www.icgst.com

# Arabic Full Text Diacritization using Light Layered Approach

Aya S. M. Hussein, Mohsen A. A. Rashwan, Amir F. Atiya
*Faculty of Engineering,Cairo University, Egypt*
ayasalah_89@yahoo.com - ayasalah_89@cu.edu.eg, mrashwan@rdi-eg.com, amir@alumni.caltech.edu
http://www.eng.cu.edu.eg

## Abstract

Text diacritic restoration is a very vital problem for languages that use diacritics in their orthography systems. Actually, it plays an important role for improving the performance of many NLP tasks. In this paper, we handle the problem of Arabic text diacritization; such that our system diacritizes input sequence of words both morphologically and syntactically. The operation of the system is divided into three layers; each layer handles a specific problem. To evaluate the performance of the system, we used the benchmark LDC Arabic Treebank datasets used by the state of the art systems, for the sake of fair comparison. Besides, we also used an extra test set to give an indication of the real performance of the system on any totally independent data set.
For morphological diacritization, we use both Hidden Markov Model and an external morphological analyser to achieve high accuracy as well as high coverage. The morphological diacritization WER achieved by the system on the benchmark test set is 3.7%. We also introduce the use of Random Forest for the syntactic diacritization and show how this simple and light classifiers is very effective such that it outperforms very powerful classifiers by achieving syntactical WER of 8.3%. Finally, we provide time analysis for each component of the proposed system.

***Keywords:*** *Arabic Text Diacritization, Natural Language Processing, Machine Learning.*

## Nomenclature

| | |
|---|---|
| $BAMA$ | Buckwalter Arabic Morphological Analyzer |
| $CRF$ | Conditional Random Fields |
| $DER$ | Diacritic Error Rate |
| $DNN$ | Deep Neural Network |
| $HMM$ | Hidden Markov Model |
| $LDC$ | Linguistic Data Consortium |
| $NLP$ | Natural Language Processing |
| $OOV$ | Out of Vocabulary |
| $POS$ | Part of Speech |
| $SVM$ | Support Vector Machine |
| $WER$ | Word Error Rate |

## 1 Introduction

The Arabic language belongs to a class of languages that uses some subscript and superscript signs, called diacritics, to determine the exact pronunciation of words. Generally, different diacritics on the same letters produce different words with maybe very different meanings.
However, most modern standard Arabic scripts are written without any diacritics. We have some exceptions like important religious text, or text that is targeted to the beginner learners of the Arabic language. Given these facts, the lack of diacritics does introduce an additional source of ambiguity. Native Arabic speakers find it a very easy task to deduce most of the missing diacritics and to infer the correct pronunciation and meaning of the word using the information provided in the context. However, this is not an easy task for Arabic beginner learners and, of course, for NLP applications that need the input words to be presented in their diacritized form as a preprocessing step.
In his work [11], M.Maamouri et al. showed that NLP applications can benefit from diacritizing input text as a preprocessing step. They experimented the effect of diacritizing input text as a preprocessing step before handling the main task which was parsing. The results showed that the accuracy of the parsing is improved when the input text is diacritized. The improvement was not significant because both parsing and diacritization use very similar linguistic features. However, the effect of diacritization on other NLP applications is expected to be significant, according to Maamouri. Actually, a long list of NLP applications is in need of a reliable automatic diacritization system to achieve higher accuracy. Examples are Machine Translation,

Text To Speech, Automatic Speech Recognition, and Word Sense Disambiguation.

The Arabic orthography consists of 28 different letters; out of them, 25 letters are constants while the remaining three letters represent long vowels. Besides, 8 different diacritics are used to indicate the exact pronunciation of the letters. Table 1 lists different diacritics and the corresponding pronunciation resulting from attaching the diacritic to the Arabic constant letter "ر". Each of the Arabic letters can be diacritized using one or more diacritics. The only possible combination of diacritics that can be attached to a letter is: "Shadda" and one of these diacritics: "Fatha", "Kasra", or "Dumma".

Table 1: Arabic diacritic set

| Diacritic | Diacritized Letter | Pronunciation |
|---|---|---|
| Fatha | رَ | /r//a/ |
| Kasra | رِ | /r//i/ |
| Dumma | رُ | /r//u/ |
| Tanween Fath | راً | /r//an/ |
| Tanween Kasr | رٍ | /r//in/ |
| Tanween Dumm | رٌ | /r//un/ |
| Shadda | رّ | /r//r/ |
| Sukon | رْ | /r/ |

The process of Arabic text diacritization can be divided into two types: the morphological diacritization and the syntactic diacritization. To be fully diacritized, each word has to be diacritized both morphologically and syntactically. The morphological diacritization is the process of restoring the word diacritics that depend only on the word itself and its meaning. Morphological diacritics of a certain word, with a certain definition, stay unchanged regardless the context of the word. Actually, morphological diacritics distinguish a word from other words having the same letters. Table 2 shows an example of how different morphological diacritics on the same letters give different words with different meanings.

Generally, most Arabic native speakers predict the correct morphological diacritics very easily and at a very high accuracy using the information provided in the context.

Table 2: Examples of words with the same letters "عقد" but with different morphological diacritics.

| Word | Part of Speech | Meaning |
|---|---|---|
| عَقْد | Noun | Contract |
| عِقْد | Noun | Necklace |
| عَقَدَ | Verb | Held |
| عُقِدَ | Verb | Was held |

On the other hand, syntactic diacritics are dependent on the role of the word in the sentence. It can be inferred from the grammar of the Arabic language. However, the grammar controlling the syntactic diacritics consists of a large set of syntax rules. These syntax rules have many special cases such that even recent native Arabic speakers face considerable difficulty in predicting the syntactic diacritics correctly. Table 3 shows different syntactic diacritics of the same morphologically diacritized word.

Table 3: Different Syntactic Diacritics of The Same Word: "عَقْد"

| Sentence | Role of the word "عَقْد" |
|---|---|
| هَذَا عَقْدٌ بَاطِلٌ | Subject |
| وَقَّعْتُ عَقْدَ العَمَلِ | Object |

Actually, there are many challenges facing the problem of Arabic text diacritization. First, the nature of the Arabic morphological system itself is very complex. Generally, any Arabic word can be factorized into prefix, stem, and postfix. But, actually, there can be more then just one prefix and one postfix attached to a single stem. For example, the Arabic word "فسيكفيكهم" is itself a complete sentence which can be translated into English as: so he will suffice you against them.

The second challenge is the lack of large fully annotated corpus to be used for training. For example, the training corpus we use in this work, which is the benchmark training corpus, contains only 288.000 words. Of course, this number of words is not enough to train a system for applying such a sophisticated task at a high accuracy.

The third challenge is that for some Arabic words, different possible diacritizations can exist for the same word with exactly the same meaning and the same POS tag. For example the Arabic translation of the word "the prophet" can be "الرَّسُول" or "الرَّسُول". Another example is the Arabic translation of the word "electricity" which can be either "كَهْرُبَاء " or "كَهْرَبَاء".

Another challenge is that there is no unified way to write the Arabic sentence. The sentence can start with a word of almost any POS tag with no restrictions. The subject can come before or after the verb. Furthermore, the subject or the verb can be totally omitted from the sentence and left for the reader/audience to guess them. Finally, beside all these challenges that are inherent in the nature of the Arabic language, there are some very common spelling mistakes that exist even in formal documents; for example, letters in these sets (ا أ إ آ ), (ة ه), (ي ى) are commonly replaced, incorrectly, by other letters in the same set.

In this work, we extend our work in [12] by enhancing the operations of each of the system layers, introducing the use of the Random Forest classifier for syntactic diacritization , and embedding knowledge of the Arabic syntactic grammar to further improve the performance of the syntactic diacritization.


The rest of the paper is organized as follows: section 2 presents different approaches for handling the Arabic text diacritization problem, as proposed by other researchers. Then, we give the details of our proposed system, in section 3. Next, we describe the experiments we hold to test the performance of our system and to compare it against the state of the art systems, in section 4. After that, the time analysis is given in section 5. Finally, we give the conclusion and suggestion for future work in section 5.

## 2 Related Work

Researchers used several techniques for handling the problem of restoring missing diacritics. Some of these techniques are rule-based, morphological-based, statistical-based, or example-based. Regardless the variations of the details, these different techniques can fit into four main categories: (1) Dealing with the problem as a statistical machine translation problem. (2) Dealing with the problem as a sequence labelling problem. (3) Factorizing words into their basic segments using morphological analysis. (4) Hybrid of two or more of the approaches listed above. The text diacritization problem can be regarded as a statistical Machine Translation problem in which the undiacritized text is considered to be the source language text, and the diacritized text is considered to be the target language text. Elshafei et al. [7] proposed the use of HMM to predict the most probable sequence of diacritized words. The main disadvantage of their system is being of low coverage. The system fails to diacritize words that haven't been encountered in the training data. To face the problem of low coverage, Schlippe et al. [22] proposed a system that uses character level n-grams for OOV words. During the real operation of the system, every word is checked to determine whether it have been encountered during the training phase. If so, the word level n-gram model is used to determine the most probable diacritized word. Otherwise, word characters are diacritized using the character level n-gram model by choosing the most likely diacritized character given the neighboring characters.

Other researchers handled the problem as a sequence labelling problem in which every word is represented as a sequence of letters. Every letter may be tagged with any of the possible diacritics. Rashwan et al. (2014)[17] and Rashwan et al.(2015)[16] used a Deep Neural Network (DNN) based framework to restore the missing diacritics. They used a set of morphological, syntactic, and context features as input to the DNN. As a post-processing step, they introduced the use of Contention Sub-set Resolution network to enhance the performance of the syntactic diacritization. Although the deep neural network achieved very high accuracy compared to other proposed systems, the sequence labelling approach usually gives lower accuracy than other approaches. The exception is when the classifier is very powerful, like the DNN. But, the cost then is the large memory and time requirements. Zitouni et al. [25] used an approach based on maximum entropy to restore the missing diacritics. They used lexical, segment-based, and POS features for the maximum entropy classifier to label the characters with the missing diacritics. Instead of labeling each character independently, the labeling is chosen such that the conditional probability of the sequence of labeled characters is maximized. A beam search algorithm was also employed to reduce the search space.

In their system, MADA, Habash and Rambow [10]used the Buckwalter Arabic Morphological Analyzer(BAMA) to produce all the possible analyses for each input word. Then, they used a set of ten trained SVM classifiers to predict the expected POS tag features of the correct analysis. Finally, They chose the analysis that best conforms to the predicted POS tag features.

Usually, the hybrid approach combines both the machine translation approach and either the sequence labelling approach or the morphological analyser based approach to achieve high accuracy as well as high coverage. Rashwan et al.(2009) [15] introduced a two-layer system in which the first layer is used to get the diacritics of previously seen words and the second layer handles the out-of-vocabulary (OOV) words. In the first layer, analyses of words that occurred during training are retrieved and A* search is used to infer the most likely sequence of diacritized words. Then OOV words are passed into a factorizing module, which consists of a morphological analyser and a POS tagger, to get their possible analyses.

Ananthakrishnan et al. [1] adopt both statistical and knowledge-based approaches to solve the diacritization problem. In training phase, they built a statistical language model using Treebank data corpus. During the diacritization, the system at first uses word-level trigram model to obtain the most probable diacritization. But, if the word is OOV, the system gets all possible morphological analyses of the word using the BAMA morphological analyzer, and then it uses character-level 4-gram model to rank the possible dicacritizations of the word.

In [12], we introduced a hybrid technique that uses HMM and morphological analyzer to restore the morphological diacritics. Then, word feature vectors are built using some morphological and lexical features. A previously trained CRF classifier is then used to predict the syntactic diacritics.

## 3 Proposed Approach

We propose an approach consisting of 3 layers, which is an extension to the work we proposed in [12], in which each layer takes a step towards the complete diacritization of the input sequence of words. We first


handle the morphological diacritization of the input sequence, then we handle the syntactic diacritization of the morphologically diacritized sequence. The first two layers tackles the morphological diacritization of the input sequence. While the first layer is responsible for morphologically diacritizing words that have occurred during the training phase using first-order Hidden Markov Model (HMM), the second layer handles the morphological diacritization of the OOV words by making use of an external Arabic Morphological Analyser. After that, the third layer is there to handle the syntactic diacritization of the morphologically diacritized sequence by making use of the Random Forest classifier. Figure 1 shows the components of our approach and the interactions between these components. The details of each of the three layers are given in the following subsections.

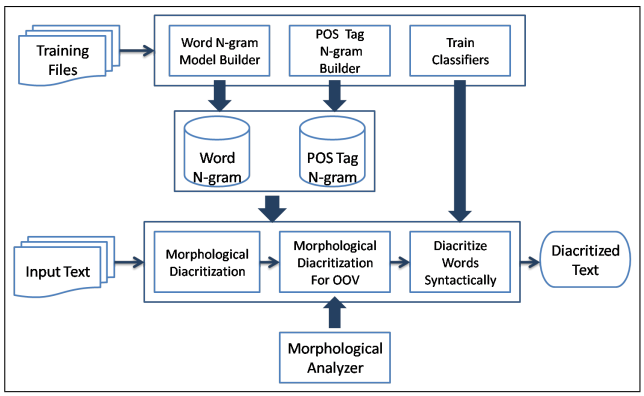


Figure 1: System Architecture

## 3.1 Morphological Diacritization of Previously Seen Words

Input words are fed into the first layer sentence by sentence. Each word in the input sentence is checked to determine whether it has been previously seen in the training data or not. If it's been previously seen, the different diacritizations of this word together with their unigrams are retrieved from the training database. If the word hasn't occurred during training, it is processed again in this layer as we begin to ask whether a variant of this word has occurred. We do so using simple letter normalization, excluding some parts of the word like prefix and postfix, and adding some of the allowed prefix and postfix to the word, then we check to see whether any of these variants of the word has occurred during training or not. If any of them has occurred, its diacritized forms are retrieved and the corresponding POS tags are formed as a combination of POS tag of the retrieved words with the POS tag(s) of the prefix and/or the postfix. Otherwise, if none of the word and its variants is found, the word is left unchanged and marked as OOV.

Then, to select the best probable sequence of morphologically diacritized words, we retrieve word bigrams from the database and apply first order HMM. To avoid the problem of zero and undefined probabilities resulting from OOV words, we use add-delta smoothing with $\delta = 0.05$. Upon selecting the words' morphologically diacritized form, POS tag bigrams are used to select the corresponding POS tags for the morphologically diacritized words. The flowchart describing the operation of layer 1 is shown in figure 2.

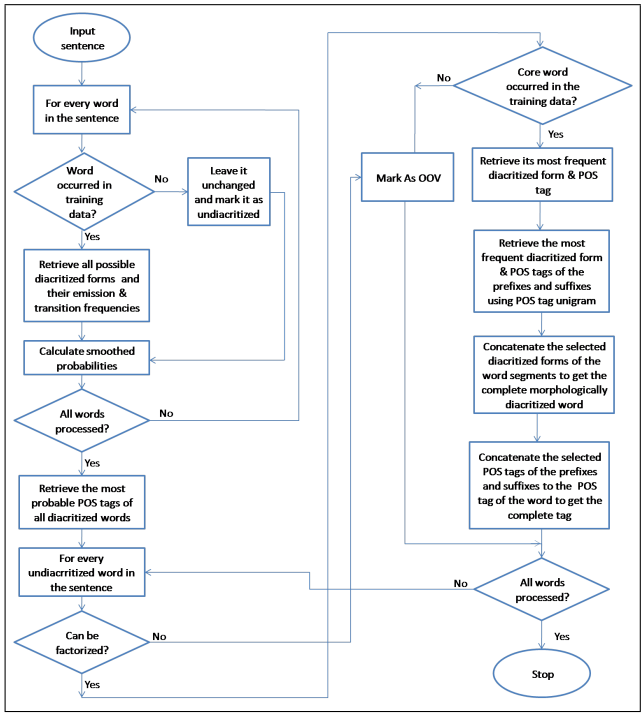


Figure 2: Flowchart of layer 1

## 3.2 Morphological Diacritization of OOV Words

The output of the first layer enters the second layer with the hope of getting the morphological diacritics for OOV words. As shown in figure 3, in this layer, we make use of the Buckwalter Arabic Morphological Analyser (BAMA) to get possible analyses for input words. The output of the analyser is a list of possible analyses for the input word. The analyses are sorted descendingly according to the frequency of the word. Each analysis consists of the diacritized form of the word and its corresponding POS tag.

Words are fed into the morphological analyser to get their analyses. If the word is previously diacritized in the first layer, we use the output of the morphological analyser only to extract some features. These features are: 1- Can the word be syntactically diacritized with any of the "Tanween" diacritics? 2- Is it common to omit the syntactic diacritic of the word?

On the other hand, if the word is OOV, the most prob-

able analysis is first chosen, and then these features are extracted. To select the most probable analysis, we make use of information gained during training about the expected POS tag as well as information provided by the morphological analyser about the relative frequency of different possible analyses. We used the following equation to calculate the probability of each analysis:

$$P(A_{i,j}) = \lambda.P(pos_{i,j}, pos_{j-1}) + (1-\lambda).\frac{1}{i+1}.N \quad )(1$$

Such that:

$A_{i,j}$ is the $i^{th}$ analysis in the output list of the morphological analyser for the word at position j, $pos_{i,j}$ is the POS tag of the analysis $A_{i,j}$, $pos_{j-1}$ is the selected POS tag of the word at position j-1, $\lambda$ is the linear interpolation parameter, $1 > \lambda > 0$, and N is a normalisation factor, $N = \frac{1}{\sum \frac{1}{i+1}}$

Once the analysis is selected, the corresponding diacritization and POS tag are assigned to the OOV word.

### 3.3 The Syntactic Diacritization

The task of the third layer, is to predict the syntactic diacritics of the input words. The input of this layer is the morphologically diacritized words that are annotated with their POS tags, as predicted by the two previous layers. Before we start diacritizing a word syntactically, we first check its POS tag. Using the POS tag of the word, we decide how the syntactic diacritization of this word should be handled. We classify every input word into one of three classes based on its annotated POS tag. The three classes are :

(1) Words annotated with noun, adjective, adverb, or number POS tags. (2) Words annotated with a POS tag of verbs in the present tense form used for singular subject. (3) Words of any other POS tag like prepositions, conjunctions, pronouns, verbs in the past tense form, and verbs used for plural subject.

We make this classification because the rules for diacritizing words in the same class are somewhat similar but different from the rules used for diacritizing words in other classes. Generally, words in the first class can be syntactically diacritized with any diacritic except "Shadda". But, the process of diacritizing these words is based on a large number of complex rules. Actually, the restoration of syntactic diacritics of words belonging to this class is much more complicated than the restoration of syntactic diacritics of words belonging to other classes. That's why, we use machine learning classifier to syntactically diacritize these words.

To predict the syntactic diacritics of words of class 1, we first need to build the feature vector. Beside the features extracted from the morphological analyzer in layer 2, some other features are extracted for each word to help predict its syntactic diacritic. We use a

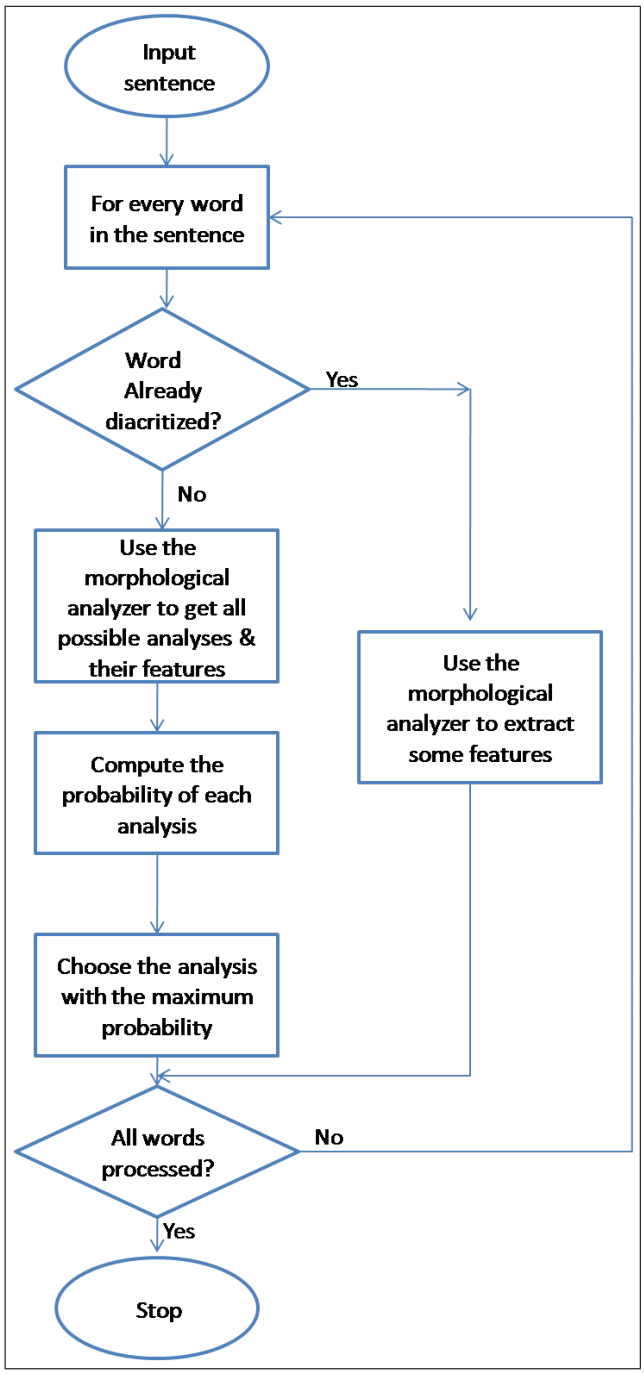


Figure 3: Flowchart of layer 2

combination of morphological, lexical, POS tag, and context features to build the feature vector. Then, we subject the feature vector to a previously trained Random Forest classifier. The Random Forest, we used, consists of 30 decision trees in which every decision tree votes for one of the candidate diacritics and the final decision, of the Random Forest as a whole, is taken as the diacritic with the highest number of votes. We used the implementation of the Random Forest that is provided in the open source "algLib" c# library [2].

The reason why we choose the Random Forest classifier for the syntactic diacritization task is that it is a rule based classifier in which each tree corresponds to a sequence of if-then-else checks. This conforms to the way in which people with strong Arabic grammar background predict the syntactic diacritics. Although people have the advantage that they make use of the semantics to further help them predict the syntactic diacritics at high accuracy, our technique still tries to mimic the way people perform this task but with much more limited sources of information.

As for words belonging to the second class, they can have any syntactic diacritic except "Shadda" and the three "Tanween" diacritics. There are a small set of easy rules that are used to select the correct syntactic diacritic for those words. Thus, words belonging to this class can be syntactically diacritized using a simple rule based classifier represented as a sequence of if-then-else statements. These if-then-else statements are directly inferred from the grammar that controls the syntactic diacritization of these words.

The syntactic diacritization of words belonging to the third class is a very trivial task. Actually, each of these words has a fixed case-ending diacritic which doesn't change in any context. So, the syntactic diacritic can be simply retrieved from the training database. The sequence of operations of layer 3 is summarized in figure 4. An example of how the system works on input sequence of words is shown in Figure 5.

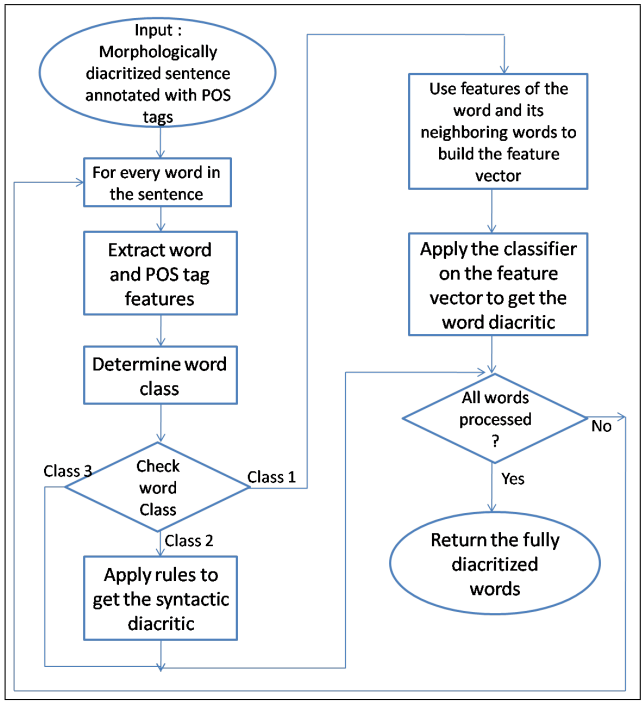


Figure 4: Flowchart of layer 3

# 4 Experimental Results

## 4.1 Data

We hold our experiments on LDC Arabic Treebank Part 3 which consists of about 600 articles from Al-Nahar Lebanese news agency. The words of the data set are annotated with their morphological and syntactic diacritics as well as with their POS tags. We use the same training and test data sets used by the state-of-the-art systems so as to guarantee fair comparison with them. The training dataset consists of 288K words representing articles from January 15, 2002 to October 15, 2002. On the other side, the test dataset consists of about 52K words representing articles from October 15, 2002 to December 15, 2002. Actually, training and test datasets were intentionally chosen to be chronologically non-overlapping so that the results model how the system will perform in the real world [20]. During the rest of the paper, we will refer to this test set as TestSet1.

Beside this benchmark test set of the Arabic Treebank, we also tested our system against another test set which is a collection of articles from "Aljazeera", annotated by RDI company in Egypt. The reason why we tested our system against this test set is that it's totally independent of the training data, as the articles are from different news agency and there is large chronological gap between this test data and the training data; most of the articles in this test data are published in 2012. This test data consists of 63 articles having about 30k words and 6k punctuation marks. The articles are of different topics from different categories. Their categories are: arts, economics, health, politics, sports, and varieties. We will call this test set TestSet2.

## 4.2 Results on TestSet1

Table 4 and table 5 show the comparison between our proposed approach and the state-of-the-art systems on TestSet1 in terms of the morphological diacritization WER and DER, the syntactic diacritization WER, and the complete diacritization WER.

The comparison indicates that both the approaches of Rashwan(2014) and Rashwan(2011) outperform our results in the morphological diacritization on TestSet1. However, we are still close to them, the difference is only about 0.7%. On the other hand, the

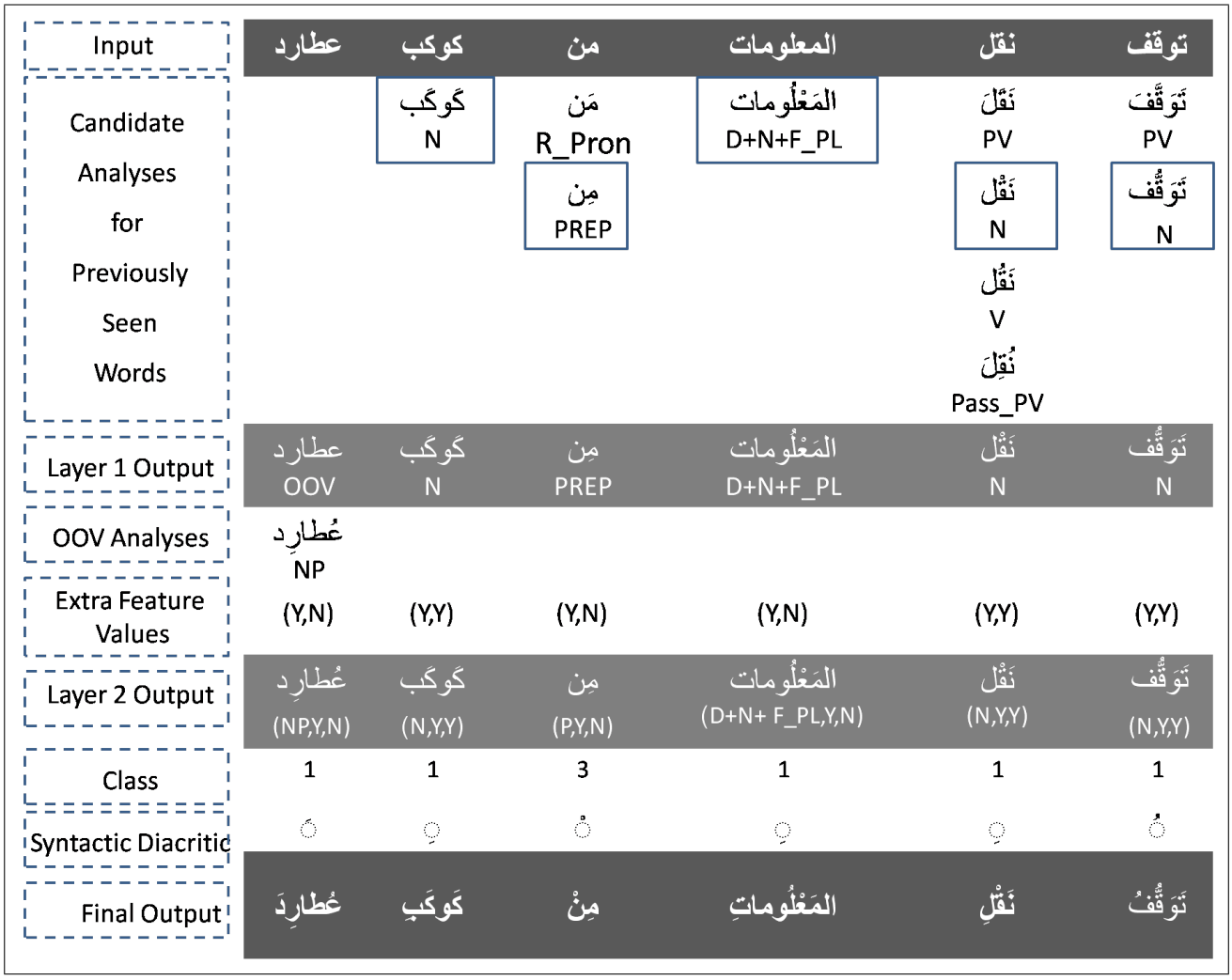


Figure 5: An example of applying the system on the input sequence "توقف نقل المعلومات من كوكب عطارد". In this figure, POS tags are abbreviated as follows: N stands for noun, R_Pron stands for relative pronoun, PREP stands for preposition, D stands for determinant, F stands for female, PL stands for plural, PV stands for past tense verv, V stands for present tense verb, and Pass stands for passive

Table 4: Comparison of the morphological diacritization WER and DER on TestSet1. The best reported results are written in bold

| Approach | Morphological WER | Morphological DER |
|---|---|---|
| Our Approach | 3.7% | 1.6% |
| Rashwan(2014) | **3%** | NA |
| Rashwan(2011) | 3.1% | **1.2%** |
| Zitouni | 7.9% | 2.5% |
| Habash | 5.5% | 2.2% |

Table 5: Comparison of the syntactic and full diacritization WER on TestSet1. The best reported results are written in bold

| Approach | Syntactic WER | Total WER |
|---|---|---|
| Our Approach | **8.3%** | 12% |
| Rashwan(2014) | 8.6% | **11.6%** |
| Rashwan(2011) | 9.11% | 12.5% |
| Zitouni | 10.1% | 18% |
| Habash | 9.4% | 14.9% |

comparison shows that our approach outperforms all the other techniques in terms of the syntactic WER by at least 0.3%. As for the total WER, the comparison shows that the work of Rashwan(2014), outperforms our results by only 0.4%. Figure 6 summarizes the results of our proposed approach in comparison with other approaches on TestSet1.

## 4.3 Syntactic Versus Case-Ending

The syntactic diacritic is the diacritic assigned to the last character of the core of the word, the stem, but not necessarily the last character of the word as a whole. On the other hand, the case-ending diacritic is the diacritic assigned to the last character of the word as a whole. For example, the word "يُعَلِمَهُ" has a syntactic diacritic "Fatha", but its case-ending diacritic "Dumma". If the word doesn't contain any postfix,

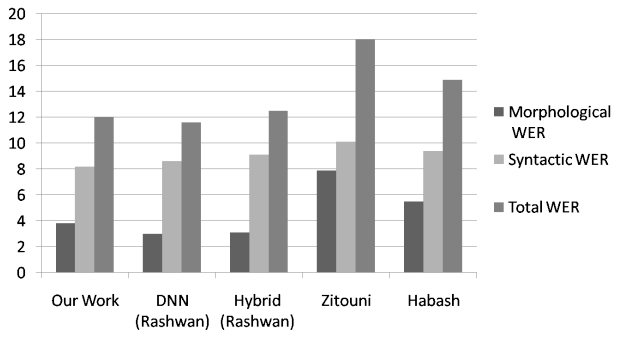


Figure 6: Graphical representation of the comparison between our system and the state-of-the-art systems on TestSet1

Table 6: The results of the system on TestSet1 using the concept of case-ending diacritic

| | |
|---|---|
| **Excluding last character WER** | 3.9% |
| **Case-ending WER** | 8.1% |

Table 7: The results of the system on TestSet2

| | |
|---|---|
| **Morphological WER** | 6.3% |
| **Syntactic WER** | 10.1% |
| **Complete WER** | 16.4% |

the syntactic diacritic and the case-ending diacritic are the same.

The syntactic WER is more accurate for expressing the powerful of the syntactic diacritization module than the case-ending WER. The syntactic WER is also exactly similar to the way people consider words correctly syntactically diacritized or not. Actually there is no real meaning for the case-ending WER in real life.

However, we noticed that some researchers reported the results of the case-ending WER instead of the syntactic WER. This also impacts the morphological WER, because the diacritic of a letter can be counted as a morphological diacritic or a syntactic diacritic( or case-ending diacritic) but not both. That's why we provide the results of our system using the concept of case-ending diacritic, here. The results of all word letters ignoring the last character diacritic as well as the results of the case-ending diacritization are reported in Table 6. Of course, the complete diacritization WER remains unchanged.

## 4.4 Results on TestSet2

As for the results of applying our system on TestSet2, they are reported in table 7.

The results show that the performance of the system is worse than its performance on TestSet1. We analysed the results and found that some words are counted as incorrect words while they should really be classified as correct by humans. This is because of the fact that some Arabic words have more than one possible diacritizations with exactly the same meaning and POS tag. 97 original Arabic words and 361 Arabic words of foreign origins exhibit this problem, in TestSet2 . We consulted the Arabic dictionaries to make sure that those original Arabic words are correct. The words "وَزارَة" , "وِزارَة" are examples of such words. As for the Arabic words from foreign origins like the words "أُغسطُس", "أَغسطُس" which can't be found in the Arabic dictionaries, we used our knowledge to decide that they are both correct. The results of the system on this test set taking into consideration the equivalent word diacritizations are reported in table 8.

Although this problem wasn't significant when we used TestSet1, it became significant when we tried

Table 8: The results of the system on TestSet2 taking equivalent diacritizations into consideration

| | |
|---|---|
| **Morphological WER** | 5.1% |
| **Syntactic WER** | 10% |
| **Total WER** | 15.3% |

this different test set because the articles in the extra test set are annotated by different human annotators with, maybe, different cultures or backgrounds.

# 5 Time Analysis

One main advantage of our system is its very competitive time requirements that make it a very powerful candidate for online operations. We will present time requirements, for both training phase and testing phase, in terms of the time complexity , in the following subsections.

## 5.1 Offline Operation Time Requirements

During the offline operation, we build the system, calculate all system parameters, and train system modules. Thus, the time complexity for training the whole system is the summation of time complexities of the three layers.

The time complexity of training the module of the first layer which uses first order HMM is the time for calculating unigram and bigram frequencies which is $O(N * logN)$; such that N is the number of words in the training data. For the second layer, no more parameters need to be calculated. Finally, to train the Random Forest in the third layer, the time complexity is $O(m * k * N * logN)$; such that m is the number of features and k is the number of decision trees in the Random Forest.
Thus,the time complexity of training the whole system is: $O(N * m * k * logN)$.

## 5.2 Online Operation Time Requirements

To calculate the total time complexity of the system in its online operation, we will sum up the time complexities of the three layer modules. The first layer uses first order HMM which has time complexity of $O(n * |S|^2)$; such that n is the length of the input sentence and $|S|$ is the number of different diacritizations, that occurred during training, for each word. In the second layer, the main operation is calculating the probability for each analysis so as to select the best probable analysis for a word form. Calculating the probabilities of h analyses of a word takes O(h) time complexity. So, the complexity of this module for input sentence of length n is $O(n * h)$. Finally, the last module uses Random Forest to get the most probable syntactic diacritic. Using Random Forest for a single word takes $O(k * log(N))$ where N is the number of training samples and k is the number of decision trees in the Random Forest. Hence the complexity of this module for an input sentence of length n is $O(n * k * log(N))$. Thus, the time complexity of the online operation of the system as a whole is: $O(n * (|S|^2 + h + k * log(N)))$. However, given the fact that S, h, k, and N are constants after the system has been built and trained, the overall complexity of the online operation becomes O(n).

# 6 Conclusion and Future Work

In this work, we proposed a multi-layer technique for handling the problem of Arabic Text Diacritization. The technique predicts both the missing morphological and syntactic diacritics at high accuracy. It uses first-order HMM as well as BAMA for morphological diacritization to achieve both high accuracy and high coverage. The WER of the morphological diacritization achieved by the system is 3.7% which is very competitive to the state-of-the-art techniques.
We introduced the use of the Random Forest classifier for the syntactic diacritization and showed that this very light classifier achieves very low syntactic WER, of just 8.3%. Our approach is much lighter than the best performing system, which uses Deep Neural Networks. However, the speed issues related to Deep Neural Networks make us a powerful rival for both online and offline operations.
As a future work, we may try to use a context sensitive morphological analysis component with the hope of improving the accuracy of OOV words.

# References

[1] Sankaranarayanan Ananthakrishnan, Shrikanth Narayanan, and Srinivas Bangalore. Automatic diacritization of arabic transcripts for automatic speech recognition. In *Proceedings of the 4th International Conference on Natural Language Processing*, pages 47–54, 2005.

[2] S Bochkanov and V Bystritsky. Alglib project.

[3] Leo Breiman. Random forests. *Machine learning*, 45(1):5–32, 2001.

[4] Mona Diab, Kadri Hacioglu, and Daniel Jurafsky. Automated methods for processing arabic text: from tokenization to base phrase chunking. *Arabic Computational Morphology: Knowledge-based and Empirical Methods. Kluwer/Springer*, 2007.


[5] Yousif A El-Imam. Phonetization of arabic: rules and algorithms. *Computer Speech & Language*, 18(4):339–373, 2004.

[6] Tarek A. El-Sadany and Mohamed A. Hashish. An arabic morphological system. *IBM Systems Journal*, 28(4):600–612, 1989.

[7] Moustafa Elshafei, Husni Al-Muhtaseb, and Mansour Alghamdi. Statistical methods for automatic diacritization of arabic text. In *The Saudi 18th National Computer Conference. Riyadh*, volume 18, pages 301–306, 2006.

[8] Raymond G Gordon Jr. Ethnologue: Languages of the world, dallas, tex.: Sil international. *Online version: http://www. ethnologue. com*, 2005.

[9] Nizar Habash and Owen Rambow. Arabic tokenization, part-of-speech tagging and morphological disambiguation in one fell swoop. In *Proceedings of the 43rd Annual Meeting on Association for Computational Linguistics*, pages 573–580. Association for Computational Linguistics, 2005.

[10] Nizar Habash and Owen Rambow. Arabic diacritization through full morphological tagging. In *Human Language Technologies 2007: The Conference of the North American Chapter of the Association for Computational Linguistics; Companion Volume, Short Papers*, pages 53–56. Association for Computational Linguistics, 2007.

[11] Mohamed Maamouri, Ann Bies, and Seth Kulick. Diacritization: A challenge to arabic treebank annotation and parsing. In *Proceedings of the Conference of the Machine Translation SIG of the British Computer Society*, 2006.

[12] Aya S. Metwally, Mohsen A. Rashwan, and Amir F. Atiya. A multi-layered approach for arabic text diacritization. In *2016 IEEE International Conference on Cloud Computing and Big Data Analysis (ICCCBDA)*, pages 389–393, July 2016.

[13] Rani Nelken and Stuart M Shieber. Arabic diacritization using weighted finite-state transducers. In *Proceedings of the ACL Workshop on Computational Approaches to Semitic Languages*, pages 79–86. Association for Computational Linguistics, 2005.

[14] Arfath Pasha, Mohamed Al-Badrashiny, Mona Diab, Ahmed El Kholy, Ramy Eskander, Nizar Habash, Manoj Pooleery, Owen Rambow, and Ryan M Roth. Madamira: A fast, comprehensive tool for morphological analysis and disambiguation of arabic. In *Proceedings of the Language Resources and Evaluation Conference (LREC), Reykjavik, Iceland*, 2014.

[15] M Rashwan, Mohammad Al-Badrashiny, Mohamed Attia, and S Abdou. A hybrid system for automatic arabic diacritization. In *The 2nd International Conference on Arabic Language Resources and Tools*, 2009.

[16] Mohsen Rashwan, Ahmad Al Sallab, Hazem M Raafat, Ahmed Rafea, et al. Deep learning framework with confused sub-set resolution architecture for automatic arabic diacritization. *Audio, Speech, and Language Processing, IEEE/ACM Transactions on*, 23(3):505–516, 2015.

[17] Mohsen AA Rashwan, Ahmad A Al Sallab, Hazem M Raafat, and Ahmed Rafea. Automatic arabic diacritics restoration based on deep nets. *ANLP 2014*, page 65, 2014.

[18] Mohamed Refaat, M Rashwan, and Nevin Darwish. Naïve bayesian modifications applied for automatic diacritization of arabic text. Master's thesis, Cairo University, 2011.

[19] Tim Schlippe, ThuyLinh Nguyen, and Stephan Vogel. Diacritization as a machine translation problem and as a sequence labeling problem. In *Proceedings of the Eighth Conference of the Association for Machine Translation in the Americas (AMTA), Hawai'i, USA*, 2008.

[20] Imed Zitouni, Jeffrey S Sorensen, and Ruhi Sarikaya. Maximum entropy based restoration of arabic diacritics. In *Proceedings of the 21st International Conference on Computational Linguistics and the 44th annual meeting of the Association for Computational Linguistics*, pages 577–584. Association for Computational Linguistics, 2006.



# Biographies

**Aya S. M. Hussein** got her B.Sc. in Computer Engineering from Cairo University, Egypt in 2011. In 2015 she received her M.Sc. in Computer Engineering from Cairo University. Her research interests are mainly: Natural Language Processing, Machine Learning, Big Data, and Optimization.

**Prof. Mohsen Rashwan** received the B.Sc. and M.Sc. degrees in electronics and electrical communications from Cairo University, another M.Sc. degree in systems and computer engineering from Carleton University, Canada, and the Ph.D. degree in electronics and electrical communications from Queen's University, Canada. He currently serves as an Emeritus Professor in the Department of Electronics and Electrical Communications, Faculty of Engineering, Cairo University, and as the R&D Director of RDI Arabic Language Technologies (HLT) research lab, (www.RDI-Eg.com). He has shared and led several national and international mega projects, like FP7 projects on Arabic HLT NEMLAR and MEDAR, and as a Co-PI in the Egyptian Data Mining Center of Excellence. He has cofounded ALTEC (an NGO to serve the Arabic language Technologies; www.ALTEC-Center.org).

**Prof. Amir F. Atiya** is currently Professor of Computer Engineering Department, Cairo University. He Was born in Cairo, Egypt. He received his B.S. degree in 1982 from Cairo University (Egypt), and the M.S. and Ph.D. degrees in 1986 and 1991 from Caltech, Pasadena, CA, all in electrical engineering. He held positions in academia, as well as several positions in financial firms. From 1997 to 2001 he was a Visiting Associate at Caltech. On leave from Cairo University, he recently held research positions in the firms Simplex Risk Management, Hong Kong, Countrywide Corporation in Los Angeles, and Dunn Capital Management, Florida. Currently, he is a Research Scientist with Veros Systems, Texas.